J. F. Medina-Lee, A. Artuñedo, J. Godoy, and J. Villagra, “Reachability Estimation in Dynamic Driving Scenes for Autonomous Vehicles,” in 2020 IEEE Intelligent Vehicles Symposium, 2020.

Reachability Estimation in Dynamic Driving Scenes for Autonomous Vehicles

Juan Felipe Medina-Lee, Antonio Artuñedo, Jorge Godoy, Jorge Villagra

***Abstract*— Autonomous vehicles will find an infinite number of possible scenarios while driving in urban environments and need to react in a proper manner. For that reason, it is important to have algorithms that can propose driving alternatives for different type of scenarios in a global and unified way instead of using rule-based algorithms which depend on the driving scene. This paper presents a reachability estimation algorithm designed to obtain a safe and comfort-optimized trajectory set for different driving scenarios. First, a finite number of path candidates are created using Bézier curves. Then, all valid path candidates are combined with the reachable sets of dynamic obstacles to generate speed profiles consistent with safety and comfort requirements. The output of this algorithm would allow a decision-making strategy to select the optimum candidate depending on different criteria.**

I. INTRODUCTION

Autonomous driving developments are increasing exponentially in the recent years in both industrial and research fields. Most recent Advanced Driver Assistance Systems (ADAS) developed by car manufacturers have raised the autonomy level of the vehicles, and current research aims for a SAE level of 4 or higher [1]. Higher autonomy levels represent a big research challenge because they require algorithms capable of handling complex scenarios in dynamic environments with little to none human intervention [1].

Many traffic-based trajectory planning algorithms are designed for specific driving scenarios like lane changes [2], intersections [3], high-speed roads [4] or obstacle-free scenarios [5]. This kind of solutions are important to understand possible autonomous driving situations, but using them in a functional driving system would imply rule-based algorithms that do not behave well in complex scenarios like urban environments [6].

More generic algorithms are found in the literature, for instance, in [7] the authors propose an architecture that generates Bézier path candidates and considers the intentions of other traffic participants to evaluate the collision probability. Hubman *et al.* [6] present a generic algorithm to find the most suitable trajectory for complex driving scenes, but it only works for one-lane scenarios.

*This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities with National Project COGDRIVE (DPI2017-86915-C3-1-R), the Community of Madrid through SEGVAUTO 4.0-CM (S2018-EMT-4362) Programme, and by the European Commission through the Projects PRYSTINE (ECSEL-783190-2) and SECREDAS (ECSEL-783119-2).

Juan Felipe Medina-Lee, Antonio Artuñedo, Jorge Godoy, Jorge Villagra are members of Autopia program, Center for Automation and Robotics (CSIC), Ctra. M300 Campo Real, km 0.200, Spain. {juan.medina,antonio.artunedo,jorge.godoy,jorge.villagra}@csic.es

Biondi *et al.* [8] remark how highly-automated ADAS in urban scenarios can encourage human operators to become complacent, under-aroused and, in turn, less responsive to traffic hazards, which compromises the safety in the driving process. This situation can be improved by implementing traded-control systems, where the human and the automation system cooperate into the driving process depending on the traffic scene. A robust reachability estimator would help a traded-control system to decide the appropriate automation level based on the quality of the candidates provided in the current context, and if the possibilities are reliable, the system-to-driver transitions of control can be handled safely.

This work presents a reachability estimation algorithm to create a set of trajectories regardless of the driving scenario and traffic participants. First, path candidates are generated using quintic Bézier curves to achieve curvature continuity along the path. Then, speed profiles are generated for each valid path in order to create a safe trajectory that complies with comfort requirements. The proposed algorithm has been tested in urban scenarios like roundabouts or multi-lane intersections, obtaining good results and meeting computing-time requirements.

The remainder of the paper is as follows: section II presents an overview of the autonomous-driving framework for autonomous vehicles in Autopia Program. Section III describes a multi-lane path generation algorithm. Section IV presents the creation of the Possible-Collision Points (PCP). Section V explains the speed profile generation algorithm and Section VI shows the results in different simulation scenarios.

II. AUTONOMOUS DRIVING ARCHITECTURE

The reachability estimator is embedded in the architecture for autonomous driving implemented in Autopia program. In this chapter, its three key inputs are highlighted: navigation corridors, dynamic occupancy grids and the reachable sets of the surrounding vehicles.

A. Navigation Corridors

Navigation corridors represent the reachable lanes for the ego-vehicle in a short-term time horizon. They may be generated either from a digital map complying with *OpenStreetMaps*(OSM) standard, as presented in [9]; or from a Lanelet2 digital map, as in [10]. The navigation corridors are described with a drivable area, a centreline and a route priority. The route priority is a relative value that compares the *distance-to-destination* of each navigation corridor in the current scenario. The navigation corridor with

the lowest *distance-to-destination* will have the greatest route priority, and the rest will have a proportional value. It is important to mention that the centreline is optimized by using a modified version of Douglas-Peucker algorithm [11] which reduces the number of points and establishes a maximum-distance separation between consecutive points. Fig.1a shows the navigation corridors for the ego-vehicle approaching a roundabout.

B. Dynamic occupancy grid

Environment perception is addressed by computing a Dynamic Occupancy Grid (*DOG*). Traditional occupancy grids allow the representation of occupied and free-space at the same time, while *DOGs* extend this perception by estimating objects' dynamics. The *DOG* also stores relevant information such as drivable area and motion prediction of the traffic agents. The size of the *DOG* depends on the instant speed of the ego-vehicle.

C. Reachable Sets of the surrounding vehicles

In order to generate safe trajectories for the ego-vehicle, the motion of the traffic participants must be predicted [1]. To generate the motion predictions, the set of reachable states within a finite time interval $t \in [0, t_f]$ needs to be computed. The reachable sets $\mathcal{R}$ of the vehicles present on the scene are computed using Markov chains. A preliminary version of this method was presented in [10] and a refined version considering interaction awareness can be found in [12]. Each prediction $\mathcal{R}_k = \mathcal{R}([t_k, t_{k+1}])$ is plotted into an independent slice of the *DOG*. The generation of this reachable sets is beyond the scope of this paper, so $\mathcal{R}$ will be considered as a given input hereinafter, using a prediction horizon $t_f = 3s$ and time-step between predictions $\Delta t = 0.1s$.

III. PATH CANDIDATES GENERATION

A set $\mathcal{P}$ of N path candidates are generated inside the possible navigation corridors using quintic Bézier curves. This curve primitive were selected after making an extensive comparison presented in [13]. Each candidate is generated from the ego-vehicle position to a specific waypoint. First, a waypoints set $\boldsymbol{\omega}$ is created based on the centrelines of the navigation corridors. Every waypoint $\omega_i \in \boldsymbol{\omega}$ is defined with a data structure including curvature, recommended speed, distance to the ego-vehicle, route priority and number of candidates.

Since there are only N path candidates for $W = |\boldsymbol{\omega}|$ waypoints, the number of candidates assigned to every waypoint NoC_{ω_i} depends on its route priority rP_{ω_i}. Thus, the waypoints belonging to high-priority corridors will have more path candidates than the waypoints of the low-priority corridors. The assignment of the number of path candidates to a waypoint is performed as follows:

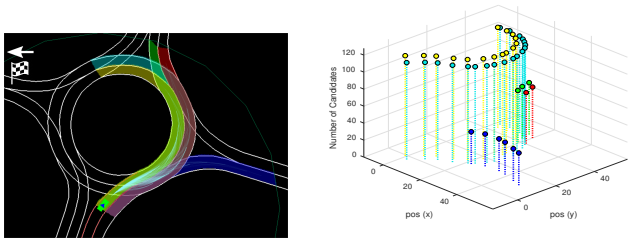
$$NoC_{\omega_i} = \frac{N \cdot rP_{\omega_i}}{\sum_{i=1}^{W} rP_{\omega_i}}, \tag{1}$$

After NoC_{ω_i} is established for the whole set $\boldsymbol{\omega}$, the path candidates set $\mathcal{P}$ is generated using the algorithm proposed in [14], which allows to impose position and curvature at both ends of the curve in order to achieve curvature continuity along the path. Then, a validity check is performed for all candidates. To consider a path candidate as valid, (i) its maximum curvature must be lower than the maximum curvature feasible by the ego-vehicle and (ii) the area occupied by the vehicle while driving along the path candidate must be inside the drivable-area. The set of all valid candidates constitute $\mathcal{P}_{valid} \in \mathcal{P}$. Fig. 1a shows the navigation corridors and the valid paths on a roundabout scenario where the goal position is to the left of the map. Fig. 1b plots the *NoC* assigned to each $\omega_i \in \boldsymbol{\omega}$. In this scenario, 4500 candidates were generated and 46 waypoints turned out after optimizing the centrelines of the navigation corridors. Hence, the parameters to calculate *NoC* are: $N = 4500$, $|\boldsymbol{\omega}| = 46$. The route priorities for the corridors are presented in table I.

Fig. 1b shows that the yellow and green corridors have a higher *NoC* while blue corridor has the lowest *NoC*. This makes sense since the goal position for the ego-vehicle is to the left of the map.

IV. POSSIBLE COLLISION POINTS

The speed profiles generated for $\mathcal{P}_{valid}$ have to avoid possible collision with dynamic obstacles and meet the comfort constraints as much as possible. In order to do that, it is necessary to estimate the spatio-temporal position of the dynamic obstacles along each valid path $p \in \mathcal{P}_{valid}$, and identify Possible Collision Points (PCP) with the ego-vehicle. In other words, PCP are a one-dimensional projection of $\mathcal{R}$ into p. This section will describe the algorithm to create the PCP for a single valid path using the driving scenario shown in Fig. 2. It is important to mention that the same algorithm is valid for any other driving scenario.



(a) (b)

Fig. 1. Navigation corridors and valid path candidates $\mathcal{P}_{valid}$ for a roundabout scenario (a) and the *NoC* for each waypoint in the navigation corridors (b)

TABLE I
ROUTE PRIORITIES FOR NAVIGATION CORRIDORS

Color of the corridor	Priority
Yellow corridor	5.00
Green corridor	2.58
Cyan corridor	4.82
Red corridor	2.52
Blue corridor	1.62

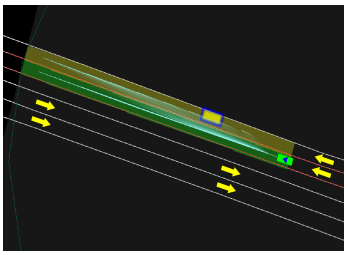


Fig. 2. Driving scenario in a two-lane highway

Fig. 2 presents a case of study in a two-lane road, where the ego-vehicle is on the left lane (green corridor) and one dynamic obstacle (car) is moving on the right lane (yellow corridor). The figure also shows $\mathcal{P}_{valid}$ and $\mathcal{R}_0 = \mathcal{R}([0s, 0.1s])$. The process to obtain the PCP is explained in algorithm 1

Algorithm 1: Possible-Collision Points generation

input : Dynamic Occupancy grid (DOG), Reachable Sets of surrounding vehicles ($\mathcal{R}[0,t_f]$), valid path set ($\mathcal{P}_{valid}$)
output: Vector of possible collision points (PCP)

```
foreach p ∈ P_valid do
    path ← InterpolateCurve(p, Δd);
    pol ← RasterizePolygon(DOG, path);
    j ← 0;
    for t ← 0 to (t_f − Δ)t do
        a ← pol ∩ (R[t, t+Δt]);
        if a is not empty then
            ς = CentersOfMass (a);
            η = NearestPoint(ς, path);
            d = DistOverPath(path, min(η));
            PCP[j]_{d,t} = d, t;
            j = j + 1;
    PCP = MovingAvgFilt (PCP, smooth);
    if PCP[0]_t > 0 then
        PCP_intr = InterpolatePCP(PCP);
        PCP = PCP_intr ∪ PCP;
```

The `interpolateCurve` function creates a poly-line *path* with equidistant segments, by interpolating a Bézier curve p using a segment length $\Delta d = 0.5m$. The `RasterizePolygon` function calculates the space occupied by the ego-vehicle after following *path* and plots it into the *DOG*, this occupancy polygon is called *pol*. Then, *pol* is intersected with the reachable sets of the surrounding vehicles at one time-step $\mathcal{R}[t, t+\Delta t]$, this intersection is called a. The `CentersOfMass` function obtains the center of mass of each cell-cluster present in a, to create ς. Then, `NearestPoint` finds the nearest segment of *path* for each $c \in \varsigma$, and appends it to η. At the end of this step, η contains the indexes of the segments in *path* which intersect with the obstacles. `DistOverPath` returns the distance-on-path for $min(\eta)$. For each PCP, it is stored both distance-on-path d and current time t. Once the PCPs are obtained, they are smoothed using a 5-order moving-average filter. Finally, if the first PCP does not start at $t = 0s$, virtual PCP are generated based on the derivative of the first 5 found PCP. This interpolation is done in order to increase the smoothness of the speed profile.

Fig. 3a illustrates algorithm 1 for one valid path of the case of study described in Fig. 2. The dark-yellow polygon that starts from the ego-vehicle is the occupancy polygon of the red path candidate, already rasterized on the *DOG*. The gradient-blue polygon on the right lane corresponds to the reachable sets ($\mathcal{R}[0,t_f]$) of the obstacle vehicle. The figure also shows three light-yellow polygons signaled with yellow, magenta and cyan marks; these polygons correspond to the variable a for $t = [1.4s, 2.1s, 2.9s]$ respectively. The center of mass of the intersecting area is used to calculate the PCPs, which are plotted in Fig. 3b.

Since the first possible collision with the obstacle occurs at $t = 1.3s$, back-propagated isochronic PCPs were generated from $t = 1.2s$ until $t = 0$. (orange circles on Fig. 3b).

V. SPEED PROFILE GENERATION

Once the dynamic obstacles have been identified for $\mathcal{P}_{valid}$, a speed profile will be generated for every $p \in \mathcal{P}_{valid}$ in order to maintain a safe distance with the obstacles while trying to satisfy comfort requirements. There are infinite ways of following a path, with different levels of comfort and safety, and the most relevant criteria to perform this task may vary following the driving context.

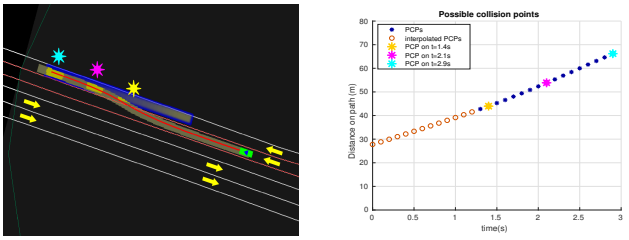
A. Obstacle-free speed profile

First, an *obstacle-free* speed profile is created based on the curvature κ of $p \in \mathcal{P}_{valid}$. In order to comply with the traffic rules and to ensure comfort inside the vehicle, this speed profile must limit both longitudinal and lateral accelerations as well as maximum speed (see table II). The final speed must also be imposed, and in our particular case, a recommended speed ω_v for the waypoint is used. Algorithm 2 presents the necessary steps to create the *obstacle-free* speed profile.

Fig. 4 shows the curvature and the *obstacle-free* speed profile of the valid path candidate in our case of study.

B. Traffic-based speed profile

The dynamic obstacles present in the scene have to be considered in order to drive safely. To that end, it has been implemented a modified version of the *Martinez&Canudas* algorithm [15], which proposes a reference speed approach for automotive longitudinal control, consistent with safety constraints and comfort specifications. An inter-distance model considers a leader vehicle and a follower vehicle (see Fig. 5). The latter is free to drive when the gap distance



(a) (b)

Fig. 3. Polygon intersection between ego-vehicle and dynamic obstacle for one path candidate (a) and its possible collision points (b)

Algorithm 2: Obstacle-free Speed profile

input : Curvature (κ_n) of $p \in \mathcal{P}_{valid}$, final speed ($(\omega_v)$)
output: *obstacle-free* speed profile for path candidate (v_n)

$v_{n,limit} \leftarrow \sqrt{\frac{a_{max,lat}}{|\kappa_n|}}$;
$v_n \leftarrow min(v_{legal}, v_{n,limit})$;
$a_n \leftarrow \frac{v_n^2 - v_{n-1}^2}{2d_p}$;
$L \leftarrow$ `Length(`κ_n`)`;
for $n \leftarrow 1$ *to* L **do**
  if $a_n > a_{max,acc}$ **then**
    $v_{n+1} \leftarrow \sqrt{v_n^2 + 2\,a_{max,acc}\,d_p}$;
    $a_n \leftarrow a_{max,acc}$;
for $n \leftarrow L$ *to* 1 **do**
  if $a_n < a_{max,dec}$ **then**
    $v_n \leftarrow \sqrt{v_{n+1}^2 + 2\,a_{max,dec}\,d_p}$;
    $a_n \leftarrow a_{max,dec}$;

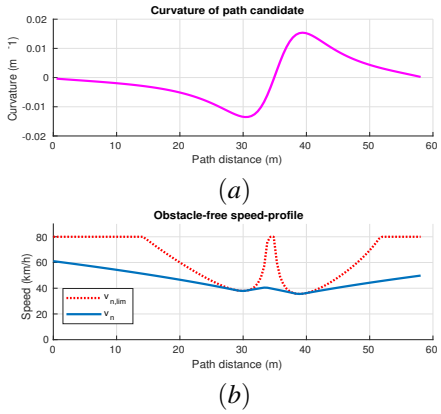


Fig. 4. Curvature of a valid path candidate (a) and its *obstacle-free* speed profile (b)

between the two vehicles $d_r = x_l - x_f$, being x_l and x_r the positions of the leader and follower, respectively, is greater than a pre-calculated threshold d_0. In this case, the follower vehicle can be considered on the *Green Zone*. If the gap between vehicles is lower than distance d_0, a non-lineal model will control the speed of the follower vehicle to keep a safe distance while meeting comfort constraints; under these circumstances, the follower vehicle is inside the *Orange Zone*, and it is the main focus of *Martinez&Canudas* algorithm. *Red Zone* occurs when $d_r < d_c$, being d_c a critical distance imposed by design; this state should never be reached.

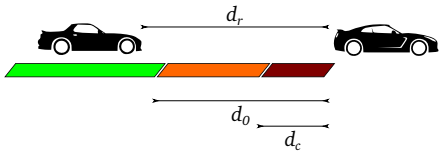


Fig. 5. Inter-distance reference model [15]

The non-linear function to establish the speed of the follower vehicle inside the orange zone can be rewritten as follows:

TABLE II
OBSTACLE-FREE SPEED PROFILE PARAMETERS

Parameter	Unit	Value
$a_{max,lat}$	(m/s^2)	1.5
$a_{max,acc}$	(m/s^2)	1.5
$a_{max,dec}$	(m/s^2)	−3.0
V_{legal}	(km/h)	80

$$\ddot{x}_f^r = -c|\tilde{d}(t)|\left[-\frac{c}{2}\tilde{d}(t)^2 + \beta - \hat{\dot{x}}_l(t)\right]$$

where:

$$c = \frac{27B_{max}^2}{8V_{max}^3} \tag{3}$$

$$\tilde{d}(t) = d_0(t) - d_r(t) \tag{4}$$

$$d_0(t) = \sqrt{\frac{16}{27}}\frac{V_{max}^2}{B_{max}} + d_c \tag{5}$$

being $\hat{\dot{x}}_l(t)$ the estimated speed of the leader vehicle. And β is the speed of the ego-vehicle when crossing the orange zone.

The algorithm needs some design parameters, defined in table III

One of the main drawbacks of *Martinez&Canudas* is that it may be too conservative when keeping a safe distance to the front-vehicle. This is because the speed of the ego-vehicle $\dot{x}_f^r$ begins to decrease once d_0 is reached; and if the algorithm is set with a $v_{max} > 72\,km/h$, then $d_0 > 50m$. That means that the ego-vehicle will begin to brake at 50m behind a leader vehicle, regardless of its current speed. Thus, the only way to reach d_c is to get to d_0 when $v_{ego} = v_{max}$ [15].

For that reason a modified version of the algorithm is proposed here, which updates the value of parameter v_{max} accordingly to v_{ego}. As a result, the ego-vehicle will start to reduce its speed when a safe brake gap distance has been reached for the current speed. The safe brake gap distance is defined as the minimal inter-distance to avoid a rear-end collision under "unpredictable" actions of the preceding vehicle. Algorithm 3 details the process to generate the dynamic speed profile for a valid path candidate.

X_{ego} represents the position of the ego-vehicle along the path. d_p is the distance between consecutive nodes on the path. It is important to mention that under no circumstances the gap distance between the ego-vehicle and the leader vehicle can be lower than d_c; if that happened for any dynamical constraints, the candidate would no longer be

TABLE III
DESIGN PARAMTERS FOR *Martinez&Canudas* ALGORITHM

Parameter	Unit	Description
B_{max}	(m/s^2)	Maximum braking acceleration
V_{max}	(m/s^2)	Maximum allowed speed
d_c	(m)	Minimum allowed inter-distance

Algorithm 3: Dynamic-obstacles speed profile

input : Possible-collision points (PCP), current ego-vehicle speed (v_{ego}), *obstacle-free* speed profile (v_n), path candidate ($\mathcal{P}_{valid,i}$)
output: Speed Profile Along Path (v_{dyn})

$orangeZone \leftarrow -1$;
$x_{ego} \leftarrow 0$;
$t \leftarrow 0$;
$v_{dyn}[0] = v_{ego}$;
$n \leftarrow 1$;
while $x_{ego} <$ `Length`$(\mathcal{P}_{valid,i})$ **do**
  $j \leftarrow$ `FindFirst`$(PCP[t > t])$;
  $x_l \leftarrow$ `LinInterp`$(PCP[j], PCP[j-1], t)$;
  $d'_0 \leftarrow \sqrt{\frac{16}{27}\frac{v_{dyn}[n-1]^2}{B_{max}}} + d_c$;
  $d_r \leftarrow x_l - x_{ego}$;
  if $d_r < d_c$ **then**
    $invalidCandidate \leftarrow true$;
    return;
  $\tilde{d'} \leftarrow d'_0 - d_r$;
  if $\tilde{d'} \geq 0$ **then**
    if $orangeZone \neq 1$ **then**
      $(c, \beta, d_0) \leftarrow$ `setInterDistParams`$(v[n-1], d_r)$;
    $orangeZone \leftarrow 1$;
    $\tilde{d} \leftarrow d_0 - d_r$;
    $v_{dyn}[n] \leftarrow -\frac{c}{2}\tilde{d}^2 + \beta$;
  else
    $orangeZone \leftarrow 0$;
    $a_{raw} \leftarrow$ `getFreeAccel`$(\tilde{d'})$;
    $v_{dyn}[n] \leftarrow \sqrt{v_{dyn}[n-1]^2 + 2a_{raw}d_p}$;
  $v_{dyn}[n] \leftarrow min(v_{dyn}[n], v_{(n,x_{ego})})$;
  $t \leftarrow t + \frac{d_p}{v_{dyn}[n]}$;
  $x_{ego} \leftarrow x_{ego} + d_p$;

considered as *valid*. `FindFirst` function returns the index of the first PCP with a collision-time higher than the elapsed time t. `LinInterp` uses the value of t to make a linear interpolation of the position between two consecutive PCP, so that the speed profile can be smoother/more-realistic. `SetInterDistParams` establishes the value of $(c, \beta \; and \; d_0)$ depending on the value of *orangeZone*; On the one hand, if $orangeZone = 0$, then $V_{max} \leftarrow v_{dyn}[n-1]$ and the value of c and d_0 are computed using equations (3) and (5) respectively. On the other hand, if $orangeZone = -1$ it means that the ego-vehicle started inside the orange zone, so optimum values of $(c, \beta \; and \; d_0)$ have to be found for the current situation by evaluating different speeds $\{v_{d_0} : v_{ego} < v_{d_0} < V_{legal}\}$ and choosing the minimum value of c. Finally, `getFreeAccel` returns an acceleration value $\{acc_{free} : 0 < acc_{free} < a_{max,acc}\}$ for driving in the green zone.

Fig. 6a shows the position evolution of the vehicles in the case of study. The green/blue dots represent the PCP; the red dots are the evolution of the ego-vehicle position along the path. The purple line displays the safe brake gap distance according to $v_{dyn}[n]$. Fig. 6b shows the *obstacle-free* speed profile, the *traffic-based* speed profile, and the acceleration profile.

The position evolution of Fig. 6a shows that the safe brake gap is maintained during the first 1.6s. The gap is bigger in the last section of the sequence because the *obstacle-free* speed is lower than the *traffic-based* speed. It is important to mention that the acceleration profile is smooth, and the

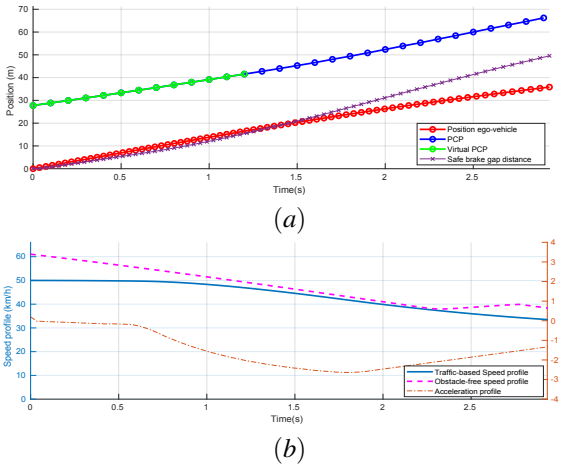


Fig. 6. Position evolution of the ego-vehicle (a) and Traffic-based speed profiles (b).

lower values are mostly found when braking due to lateral acceleration limit.

VI. RESULTS

The performance of the reachability estimator has been evaluated using two different scenarios: a roundabout and a two-lane intersection. Different trajectories of the whole set proposed by the system in both scenarios are analyzed in this section.

A. Roundabout scenario

Fig. 7 shows the first testing scenario, where the ego-vehicle is approaching to a roundabout, and two other vehicles are on the scene. Two valid paths (magenta and yellow paths on Fig. 7) were selected in order to show the speed profiles generated for each. The initial speed for the ego-vehicle is 25 km/h, for the Yellow vehicle is 30 km/h and for the Cyan vehicle is 10 km/h.

Fig. 8 shows the results for magenta path candidate. The position evolution shows that PCP were generated for the yellow vehicle, starting at $t = 0.8s$; it also displays that the safe brake gap was not reached at any point. This is due to the maximum allowed speed imposed by the *obstacle-free* speed profile (magenta dashed line on Fig. 8b). The acceleration along the path was bounded to $a_n \in [-0.43m/s^2, 0.15m/s^2]$ and it has no discontinuities (Fig. 8b).

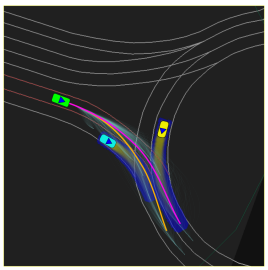


Fig. 7. Roundabout scenario with two obstacle-vehicles

Fig. 10 shows the expected position for the vehicles in the roundabout scenario for $t = [0s, 1s, 2.6s]$. The color of the direction triangles inside each vehicle indicates the corresponding time-step. Fig 10a shows the position evolution after following the magenta path candidate and Fig 10b shows it for the yellow path candidate. The time-steps are also plotted on Fig. 8 and Fig. 9.

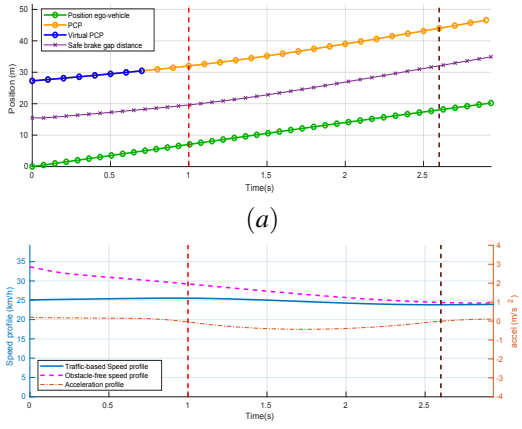
Fig. 9 shows the results for the yellow-path candidate. In this case, PCP were generated for the cyan vehicle starting at $t = 0.9s$. The gap between the ego-vehicle and PCP was also not reached because as in magenta path. The acceleration along this path was bounded to $a_n \in [-1.5m/s^2, 0.3m/s^2]$.

B. Double-lane intersection scenario

The second testing scenario is shown in Fig. 11. This time, the ego-vehicle is heading to a four-way intersection on a double-lane road. The scenario shows two obstacle vehicles on the scene, one of them is heading on the same direction as the ego-vehicle and the other one is approaching perpendicularly, but it is closer to the intersection than the ego-vehicle. In this case, only one path is selected for showing the results. The initial speed for the ego-vehicle is 30 km/h, for the Yellow vehicle is 25 km/h and for the Cyan vehicle is 40 km/h.

Fig. 12 shows the results for the red path candidate across the intersection. The PCP generated for this path are interesting, since they are originally only present for $t \in [1.2s, 1.6s]$, and then back propagated until $t = 0s$. The safe brake gap was never reached along the path. The results of Fig. 12b show a continuous acceleration profile bounded to $a_n \in [-2.06m/s^2, 1.5m/s^2]$.

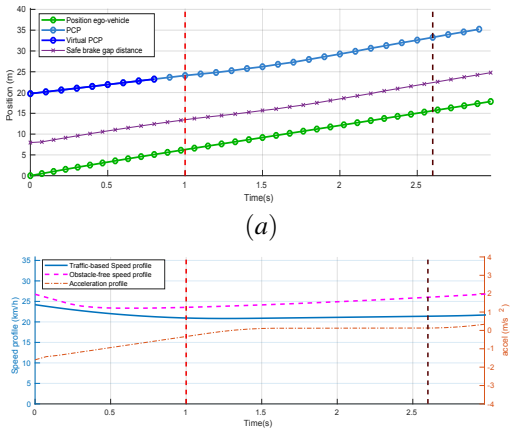
Fig. 13 shows the expected position for the vehicles in the intersection scenario for $t = [0s, 1.4s, 2.6s]$. The color of the direction triangles inside each vehicle indicates the corresponding time-step. It can be seen that by the time the ego-vehicle is crossing the intersection, the cyan vehicle has already crossed and it is in a safe position. The time-steps are also plotted on Fig. 12.



(a)

(b)

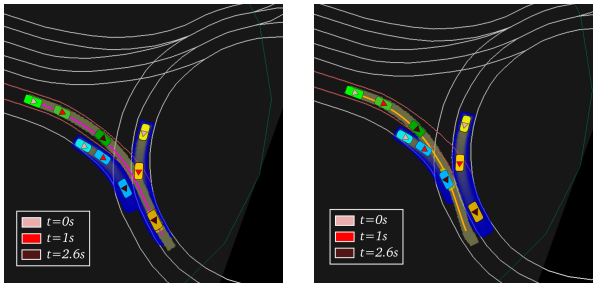
Fig. 8. Position evolution of the ego-vehicle (a) and speed profiles (b) for magenta path candidate in roundabout scenario.



(a)

(b)

Fig. 9. Position evolution of the ego-vehicle (a) and speed profiles (b) for yellow-path candidate in roundabout scenario.



(a) (b)

Fig. 10. Expected position evolution of the vehicles for the magenta path (a) and yellow path (b) in the roundabout scenario

Both acceleration and jerk affect the ride quality on an vehicle. According to the authors in [16], trajectories bounded to $|a_{max}| \leq 2m/s^2$ and $|j_{max}| \leq 0.9m/s^3$ are comfortable for the passengers inside the vehicle. Table IV shows the maximum jerk values for the three trajectories analyzed on this section. The values show that the comfort levels proposed in [16] are met, except for the trajectory of the intersection scenario, where the jerk is a little higher than recommended. This is due to the fact that safety has a greater priority in the speed profile generation, and comfort requirements are only met when safety on the scene allows it.

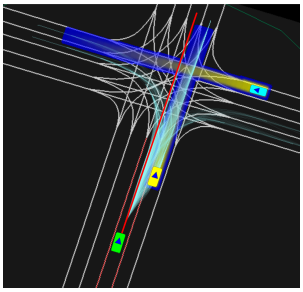


Fig. 11. Double-lane intersection scenario with two obstacles

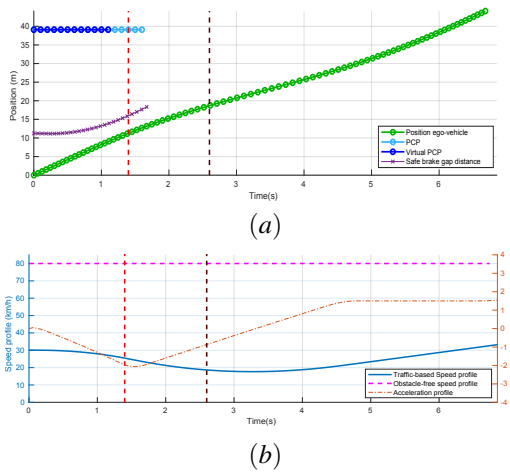


Fig. 12. Position evolution of the ego-vehicle (a) and speed profiles (b) in the intersection scenario.

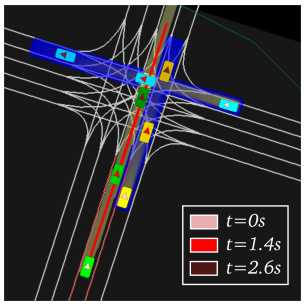


Fig. 13. Expected position evolution of the vehicles in the intersection scenario

C. Time-execution results

Table V shows the average execution times *per candidate* for independent tasks. All experiments were conducted on a PC with an Intel Core i7-7700HQ 2.8GHz processor and 8GB RAM.

In average, it takes a third of a millisecond to generate a complete valid trajectory candidate.

TABLE IV

COMFORT VALUES FOR TRAJECTORIES

Experiment	Acc. min.	Acc. Max.	J. min.	J. Max.
Roundabout: *Magenta*	−0.43	0.15	−1.06	0.87
Roundabout: *Yellow*	−1.50	0.31	0.00	0.94
Intersection: *Red*	−2.06	1.53	−1.61	1.12

TABLE V

COMPUTING TIMES (MS) PER CANDIDATE

Task	Roundabout	Intersection	Average
Create Occupancy polygon	0.114	0.132	0.123
Generate PCP	0.177	0.117	0.147
Computing speed profiles	0.054	0.070	0.062
Total time per candidate	0.344	0.319	0.332

VII. CONCLUSIONS

In this paper, it is presented a reachability estimator that generates safe and comfortable trajectories for different dynamic environments with a global algorithm. The performance of the estimator was tested in complex scenarios like roundabouts, lane changes and intersections, with satisfactory results in both safety and comfort requirements. Computation time results show that the algorithm can be suitable for real-time applications.

The trajectory sets generated by the reachability estimator presented in this work can be used by a high level decision-making algorithm to select the best driving trajectory based on any given criteria. It also provides relevant information to establish the most suitable SAE automation level for any given driving scene, which will be further explored in future works.

REFERENCES

[1] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," In: *IEEE Intell. Veh. Symp. Proc., no. Iv.*, pp. 1671–1678, 2017.

[2] K. Liu, J. Gong, A. Kurt, H. Chen, and U. Ozguner, "Dynamic Modeling and Control of High-Speed Automated Vehicles for Lane Change Maneuver," In: *IEEE Trans. Intell. Veh.*, 329–339, 2018.

[3] A. C. Charalampidis and D. Gillet, "Speed profile optimization for vehicles crossing an intersection under a safety constraint," In: *2014 Eur. Control Conf. ECC 2014.*, pp. 2894–2901, 2014.

[4] B. Park, Y. C. Lee, and W. Y. Han, "Trajectory generation method using Bézier spiral curves for high-speed on-road autonomous vehicles," In: *IEEE Int. Conf. Autom. Sci. Eng., vol. 2014-Janua.*, pp. 927–932, 2014.

[5] J. Villagra, V. Milanés, J. Pérez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle," In: *Rob. Auton. Syst.*, vol. 60, no. 2, pp. 252–265, 2012.

[6] C. Hubmann, M. Aeberhard, and C. Stiller, "A generic driving strategy for urban environments," In: *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC.*, pp. 1010–1016, 2016.

[7] H. Chen, X. Wang, and J. Wang, "A Trajectory Planning Method Considering Intention-aware Uncertainty for Autonomous Vehicles, " In: *Proc. 2018 Chinese Autom. Congr. CAC 2018.*, pp.1460–1465, 2018.

[8] I. Alvarez, and K. A. Jeong, "Human–Vehicle Cooperation in Automated Driving: A Multidisciplinary Review and Appraisal," In: *Int. J. Hum. Comput. Interact.*, pp. 1–15, 2019.

[9] A. Artunedo, J. Godoy, and J. Villagra, "A decision-making architecture for automated driving without detailed prior maps," In: *2019 IEEE Intell. Veh. Symp.*, 2019.

[10] J. F. Medina-Lee, V. Trentin, and J. Villagra. "Framework for motion prediction of vehicles in a simulation environment," in *XL Jornadas de Automática: libro de actas*, pp. 520–527, 2019.

[11] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature," pp. 15–28. John Wiley & Sons, Ltd, 2011.

[12] V. Trentin, J. Medina-Lee, V. Jimenez, J. Villagra, "A novel approach for motion prediction considering vehicles interactions," in *submitted in International Conference on Intelligent Transportation Systems (ITSC) 2020.*

[13] A. Artunedo, J. Godoy, and J. Villagra, "A Primitive Comparison for Traffic-Free Path Planning," in *IEEE Access, vol. 6*, pp. 28801–28817, 2018.

[14] A. Artunedo, J. Villagra, and J. Godoy, "Real-Time Motion Planning Approach for Automated Driving in Urban Environments," In: *IEEE Access, vol. 7.*, pp. 180039–180053, 2019.

[15] J. J. Martinez and C. Canudas-de-Wit, "A safe longitudinal control for adaptive cruise control and stop-and-go scenarios," In: *IEEE Trans. Control Syst. Technol., vol. 15, no. 2.*,pp. 246–258, 2007.

[16] J. Eriksson and L. Svensson, "Tuning for ride quality in autonomous vehicle: Application to linear quadratic path planning algorithm," Dissertation, Uppsala University, Uppsala, Sweden, 2015.